UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/077,520	02/15/2002	Fintan Ryan	5181-78701	3344

58467 7590 01/06/2010

MHKKG/SUN
P.O. BOX 398
AUSTIN, TX 78767

EXAMINER	
BOUTAH, ALINA A	

ART UNIT	PAPER NUMBER
2443	

NOTIFICATION DATE	DELIVERY MODE
01/06/2010	ELECTRONIC

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Notice of the Office communication was sent electronically on above-indicated "Notification Date" to the following e-mail address(es):

patent_docketing@intprop.com
ptomhkkg@gmail.com

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE BOARD OF PATENT APPEALS
AND INTERFERENCES

Ex parte FINTAN RYAN

Appeal 2009-005903
Application 10/077,520
Technology Center 2400

Decided: January 4, 2010

Before KENNETH W. HAIRSTON, ROBERT E. NAPPI, and KARL D. EASTHOM, *Administrative Patent Judges*.

EASTHOM, *Administrative Patent Judge*.

DECISION ON APPEAL

STATEMENT OF THE CASE

Appellant appeals (App. Br. 4)[1] under 35 U.S.C. § 134(a) from a final rejection of all the pending claims, claims 1-72. We have jurisdiction under 35 U.S.C. § 6(b).

We affirm.

Appellant's disclosed invention generates, on an intelligent device (i.e., a computer system), a batch configuration document by accessing a plurality of configuration files on the computer system, so that the batch file configures (i.e., tunes, optimizes) software components on the computer. As an example, a software component may have configuration files with parameters that can be tuned so that the component uses less memory, or runs faster. In prior art systems, a user accessed each configuration system to tune each software component separately on the computer. (Spec. 1: 14-18, 2: 2-17; 4: 1-15; 5: 12-17; 16: 21 to 17: 6; 33: 16-19; 18: 1-3; Figs.1, 4.)

Exemplary claim 1 follows:

1. A method for generating a batch configuration document for an intelligent device, the method comprising:

accessing a plurality of configuration files on the intelligent device, wherein each of the one or more configuration files includes configuration information for one of a plurality of software components of the intelligent device; and

generating the batch configuration document from the plurality of configuration files, wherein the batch configuration document

[1] This opinion refers to Appellant's Brief [hereinafter "App. Br."] and Reply Brief [hereinafter "Reply Br."], and the Examiner's Answer [hereinafter "Ans."].

includes the configuration information for the plurality of software components of the intelligent device;

wherein, after said generating, the batch configuration document is accessible for use in configuring the plurality of software components of the intelligent device whose configuration files were used in said [sic] generating the batch configuration document.

The Examiner relies on the following prior art references:

Mossman	US 2002/0124061	Sept. 5, 2002
Shafron	US 2003/0014479 A1	Jan. 16, 2003
Horman	US 6,785,706 B1	Aug. 31, 2004

The Examiner rejected claims 1-5, 7-18, 23-56, 60-66, 68-70, and 72 as obvious under 35 U.S.C. § 103(a) based on Horman and Mossman.[2]

The Examiner rejected claims 6, 19-22, 57-59, 67, and 71 as obvious under 35 U.S.C. § 103(a) based on Horman, Mossman, and Shafron.

ISSUES

Appellant's contentions (App. Br. 13-16) with respect to claim 1 reduce to the following issue: Did the Examiner err in finding that Horman and Mossman collectively teach accessing a plurality of configuration files

[2] The Examiner harmlessly omitted claim 70 in the rejection heading, but did include it in the statement of rejection. (Ans. 4, 14; *accord* App. Br. 18.)

on an intelligent device, the device generating a batch configuration document from the plurality of configuration files, as set forth in claim 1?[3]

Appellant's contentions with respect to the remaining claims are addressed more specifically below. In general, the issues involve whether or not the Examiner erred in finding that it would have been obvious to transfer a batch program from one intelligent device to another, generate the document or files in one language or another, initiate the software components, employ re-boot commands, employ drivers or operating systems as configured software components, and related issues.

FINDINGS OF FACT (FF)

Appellant's Description of the Related Art

1. Appellant states that intelligent devices "typically include one or more software components that are executable within the devices. Intelligent devices may include, but by no means are limited to: smart appliances, printers, . . . desktop computers, workstations, more powerful computer systems such as mainframes and high-end servers, even supercomputers." (Spec. 1: 14-18.)

2. Appellant explains that related art "[s]oftware components often have parameters that are modifiable or 'tunable' to change one or more aspects of the performance of the component." (Spec. 1: 27-28.) "Typically, when an intelligent device is initially configured, each component's configuration file(s) will contain default values for the parameters used to tune the component." (Spec. 3: 1-3.)

---

[3] Appellant groups (App. Br. 13) claims 1, 9, 10, 41, 48, 52, 66, and 68 together with arguments focusing on independent claim 1. Thus, claim 1 is selected to represent the group. *See* 37 C.F.R. § 41.37(c)(1)(vii).

3. Appellant provides examples of these related art tunable parameters:

> For example, a software application may have parameters that may be changed to adjust the amount of memory . . . [to] allow the application to run faster . . . . Another example of component parameters is system software parameters, including operating system parameters . . . . For example, a system software parameter may specify the network address of the intelligent device. . . . Yet another example of component parameters is driver parameters . . . to control some aspect of a hardware component.

(Spec. 1: 28 to 2: 7)

4. Appellant describes how, "[i]n the prior art, if one or more other intelligent devices need to be tuned to match a first device, . . . the user . . . may transfer one or more of the configuration files 106 to the one or more other devices device [sic]." (Spec. 4: 11-15; *see* prior art Fig.1).

Appellant's Disclosed Invention

5. Appellant describes the same intelligent devices as those listed above as related art. (*Compare* FF 1 *with* Spec. 14: 3-17.) In addition, Appellant states that "[i]ntelligent devices typically include one or more hardware components such as processors, memory, storage devices, and external interfaces." (Spec. 14: 15-17.)

6. "In one embodiment, the batch configuration document may be divided into sections, with one section for each configuration file. . . . The sections may also include name and location (e.g. directory) information for the configuration files." (Spec. 6: 7-11.)

Horman

7. Horman discloses a network of administered servers. The network includes an administrative control server configuring optimization changes to the administered data base servers. For example, the control server optimizes memory allocations for each member of a server group. The administered servers are grouped according to similar, but not necessarily identical configurations.

Users of administered servers within a group share the same data base and user-applications and typically have the same or a similar occupation, such as selling life insurance. (Abstract; col. 1, l. 33 to col. 2, l. 36; col. 5, ll. 22-47; col. 9, ll. 48-53.) Horman states that the invention is not limited to data base servers, but "can be applied to the database environment." (Col. 5, ll. 8-10.)

8. Grouping administered servers allows for the administration of "hundreds, if not thousands" of servers that perform the same functions. (Col. 5, ll. 33-35.) "The administration solution . . . is fully scalable." (Col. 5, ll. 40-41). Additional similar servers can be added to the group. (Col. 5, l. 37-39; col. 10, ll. 6-12.)

9. Each administrative server in a group shares the same "end-user application that runs on them, and the database definition that supports the application." (Col. 5, ll. 19-22.) A "database definition is the setup of the database system, including, but not limited to, the instance, the database manager configuration parameter values, the database design, and the database configuration parameter values." (Col. 5, ll. 23-27.) "Depending on its version of the end-user application, however, it may share its database

definition and data with only a subset of the administered servers that belong to its group." (Col. 5, ll. 60-63.)

10. "Within a group, administered servers may run different versions of the end-user application, and each version of the end-user application may require its own database definition and data. Group batches are preferably associated with a particular 'application version.' The application version represents the database definition and data that support a particular version of the end-user application." (Col. 5, ll. 49-55.)

11. To set up and maintain administered servers, Horman employs batch steps, each of which include, *inter alia*, a script. A set of batch steps specific to a group constitutes a group batch. Each group of administered servers, associated with one application, has several group batch types. Group batches serve, *inter alia*, to set up database and database manager configuration parameters and maintain the data.

Each batch step in a batch contains, *inter alia*: 1) a script (e.g., a data base command, an SQL statement, or an operating system command); 2) an execution target (e.g., a database instance, database, or operating system on the administrative server); and 3) authentication credentials (e.g., ID, password). (Col. 5, ll. 43-55; col. 8, ll. 8-30, FF 15 *infra.*)

12. A Parameterized Script Template (PST) includes sets of scripts which include commands and parameter markers. (Col. 3, ll. 6-9.) This template changes the configurations of the administered servers both as a group, and, on a member by member basis. (Col. 2, ll. 50-65 and the following):

> The PST describes the state transition that all members of the group must follow, so it uses the embedded parameter markers to abstract away member-specific attributes or characteristics.

> *The actual script that a group member must execute to effect the change of state is an instantiation of the PST wherein all parameter markers are replaced by corresponding values that [sic] particular group member's characteristics.*

(Col. 3, ll. 14-21 (emphasis added).)

"That is, *the parameters in a PST represent the attributes or characteristics that are unique to each member of the group*. In effect, the parameters allow for the existence of a general description of administrative action of the group (the PST) *by abstracting away the differences among the members of the group*." (Col. 2, ll. 59-65 (emphasis added).)

The PST uses relational tables of unique characteristic values for each group member which values are substituted for parameter markers within the PST. (Col. 4, ll. 61-66.) These relational tables store the values for parameter substitution and each have a unique identifier associated with a specific group member. (Col. 3, ll. 45-51). "The process of instantiation extracts the characteristics of a given group member from the relational tables . . . as directed by the parameter markers." (Col. 3, ll. 54-56.)

The administrative control server (11) contains an administrative control data base (12) which contains "information that configures and maintains administered servers . . . in relational tables . . . ." (Col. 6, ll. 24-32.)

13. Administered servers download and execute the group batches locally in a process called "synchronization." (Col. 6, ll. 9-14.) "When an administered server synchronizes, it reports its application version to its administrative control server before it downloads and executes the scripts associated with the group batches for the application version." (Col. 8, ll.

47-51.) Each server must be authenticated upon connection of an administered server to the administrative control data base. (Col. 9, ll. 6-7.)

14. Various actions follow authentication:

> 3. After authentication occurs, the administrative control sever checks which group the administered server belongs to, and the version of the application that the administered server is executing. The administrative control server uses this information to determine which batches the administered server should execute, and which batch steps should be executed, if any. At this time, other events may also occur:
>
> a) . . .
>
> b) *If any of the scripts in a batch to be downloaded are parameterized, the database control server instantiates the script with the values that are appropriate for the administered server.*
>
> c) When steps 3a and 3b are complete (if required), the administrative control server releases the scripts that the administered server is to execute, and the administered server downloads them.

(Col. 9, ll. 8 -27 (emphasis added).)

15. The application version usually is setup on the administered server during installation (setup batch) and maintained (update batch) thereafter. Thus, administered servers have three main types of batches associated therewith: 1) Group (includes setup, update, and clean up batches, 2) Fix, and 3) Unassigned. (Col. 11, ll. 35-65, col. 13, ll. 50-61).

Setup batches "populate . . . tables" (col. 12, l. 42) and setup the "database definition on the administered server, including schemas, tables, indexes, and any other database objects that are required. The setup batch can also be used to set configuration parameter values and to set up data replication." (Col. 12, ll. 59-64.) New batch steps appended to the setup

batch can be executed after the initial synchronization. The server executes only the appended steps if required to modify the data base definition, but executes the whole batch to maintain the data. (Col. 12, ll. 43-48.)

Update batches primarily replicate data without changing the data base structure (i.e., without changing the data base definition (e.g. row and column structure). (Col. 15, ll. 13-22.) A "typical update batch will consist of a data synchronization batch step . . . . [U]pdate batch [steps] . . . can be repeatedly executed without changing the current state or database definition of the administered server to a different state with each invocation of the step. For example, a table can be replicated multiple times without resulting in a change in the replication configuration." (Col. 13, ll. 3-11.)

"Batches can also be used to fix administered servers that either report problems, or require an adjustment." (Col. 11, ll. 24-25.)

"Batches are used to ensure that the administered servers in a group remain as similar as possible." (Col. 11, 16-17.)

16. Servers are added in staged deployments, and/or applications are tested on model servers, to ensure performance and modify as necessary. (Col. 10, ll. 38-64.) These new application versions execute copies of the prior batch with, for example, additions of indexes to tables to handle more data, or additional batch steps. Each modified batch, depending on the batch type, includes the old batch steps and the added steps. After modification, the new batch group (associated with the new tested version) is downloaded to and executed on existing group members previously executing the prior version.

Newly added servers execute the whole setup batch to synchronize, while existing servers execute steps depending on when the servers were

rolled out in relation to the modified batches. Modifications can be made on test state application versions and their associated batch groups, which are created from copies of prior production versions and associated batches. Copies of batches are made available from a test server to production servers. Any changes, including deletions can be made to test level applications and associated test batches. (Col. 7, ll. 16-22, col. 7, ll. 53-59, col. 10, l. 65 to col. 11, l. 15; col. 14, ll. 29-50; col. 15, ll. 14-49; col. 16, ll. 51-67; col. 18, ll. 18-67; col. 19, ll. 37-39.)

Shafron

17. Shafron teaches that a script "may be written in any suitable programming language . . . as a routine matter of design choice." (¶0031.) The script points to a control that handles data between the script and a server. The control can be referenced in the script "by use of its id, name or any other relevant application of the Document Object Model (DOM), as a matter of design choice." (¶0032.)

PRINCIPLES OF LAW

"[T]he examiner bears the initial burden, on review of the prior art or on any other ground, of presenting a *prima facie* case of unpatentability." *In re Oetiker*, 977 F.2d 1443, 1445 (Fed. Cir. 1992). A determination of obviousness can be based on a showing that "there was an apparent reason to combine the known elements in the fashion claimed." *KSR Int'l Co. v. Teleflex Inc.*, 550 U.S. 398, 418 (2007). Appellants have the burden on appeal to show reversible error by the Examiner in maintaining the rejection. *See In re Kahn*, 441 F.3d 977, 985-86 (Fed. Cir. 2006) ("On appeal to the Board, an applicant can overcome a rejection by showing insufficient

evidence of *prima facie* obviousness or by rebutting the *prima facie* case with evidence of secondary indicia of nonobviousness.") (citation omitted).

"[T]here must be some articulated reasoning with some rational underpinning to support the legal conclusion of obviousness." *Id.* at 988. "[W]hen a patent 'simply arranges old elements with each performing the same function it had been known to perform' and yields no more than one would expect from such an arrangement, the combination is obvious." *KSR*, 550 U.S. at 417 (citation omitted). "'Our suggestion test is in actuality quite flexible and not only permits, but *requires*, consideration of common knowledge and common sense.'" *Id.* at 421 (citation omitted).

"'The combination of familiar elements according to known methods is likely to be obvious when it does no more than yield predictable results.'" *Leapfrog Enters., Inc. v. Fisher-Price, Inc.*, 485 F.3d 1157, 1161 (Fed. Cir. 2007) (quoting *KSR*, 550 U.S. at 416).

> A reference may be said to teach away when a person of ordinary skill, upon reading the reference, would be discouraged from following the path set out in the reference, or would be led in a direction divergent from the path that was taken by the applicant. The degree of teaching away will of course depend on the particular facts; in general, a reference will teach away if it suggests that the line of development flowing from the reference's disclosure is unlikely to be productive of the result sought by the applicant.

In re Gurley, 27 F.3d 551, 553 (Fed. Cir. 1994).

ANALYSIS

Claims 1, 9, 10, 41, 48, 52, 66, and 68

While Appellant challenges (App. Br. 14) the Examiner's reliance (Ans. 5) on Mossman to teach "accessing a plurality of configuration files on the intelligent device," as required by claim 1, Horman's administrative

control server constitutes an intelligent device. As explained more fully below, Horman's batch configuration document accesses configuration files for software components, including the data base structures and data for each administered server. The batch file and the configuration files are on the intelligent device – the administrative control server. (FF 7, 11-14.)

Under an alternative interpretation of an "intelligent device," Horman teaches a computer network, particularly a network of administered servers (administered by an administrative control server). Horman also teaches that the invention generally applies to a data base environment. (FF 7.) Similarly, Appellant discloses that "[i]ntelligent devices may include, but by no means are limited to: . . . powerful computer systems such as mainframes and high-end servers, even supercomputers." (FF 1.) Appellant also discloses intelligent devices as including one or more of "storage devices" and "external interfaces." (FF 5.) Appellant's disclosed inter-connected components indicate the inclusion of network connections as part of the claimed intelligent device, such as Horman's network.

Therefore, in light of Appellant's disclosure, Horman's network reasonably corresponds to Appellant's disclosed "powerful computer system[]" "intelligent device," where Appellant's disclosure generally does not limit the claimed "intelligent device," and specifically, does not limit it to single hardware components. (FF 1, 5.) In other words, Mossman's teachings are cumulative, as Horman's control server or server network system, corresponds to or suggests, the "intelligent device" recited in claim 1.

Appellant also challenges (App. Br. 14-15) the Examiner's finding that Horman teaches "generating the batch configuration document from the

plurality of configuration files," as claim 1 also requires. The Examiner relies (Ans. 5) on Horman's teachings at columns 5 and 6. There, Horman describes, *inter alia*, groups of administered servers each accessing and executing a group batch, with batches comprising sets of scripts. (FF 10-14.)

Each group batch consists of a PST, a template that contains general and specific information for each distinct member of a specific group of similar administered servers. The template also accesses relational tables, indexes, and other data base objects, some of which have values common, and others of which have values unique, to each administered server of the group. Each administered server of a group executes its group batch containing specific values to optimize (configure) software, such as the data base configuration and maintenance, for that particular server. (FF 7, 10-15.)

These sets of either common and/or unique values for each administered server (stored in relational tables and indexed on the control server), accessed by a particular group batch of scripts, constitute a plurality of configuration files containing configuration information to generate the particular group batch steps (i.e. including parameterized scripts). (*Id.*) Each table and set of unique values also constitutes at least part of a unique configuration file for each administered server.

Horman's batch file document, comprising, *inter alia*, the PST, with access to common and unique configuration tables for each administered server and software therein, appears similar to Appellant's "divided" batch configuration document and tables (FF 6). Therefore, Horman meets the step of "generating the batch configuration files from the plurality of

configuration files, wherein the batch configuration document includes the configuration information for the plurality of software components of the intelligent device," as required by claim 1.

The accessing step of claim 1 requires accessing a plurality of configuration files on the intelligent device. Those files are accessed by the administered servers and/or the control server, e.g., when the batch file is generated during tests, during instantiation (parameter values extracted from relational tables (FF 12)), upon synchronization of each administered server, after synchronization (i.e., when the original setup file has scripts appended thereto (FF 15)), during batch modification, and during server addition, update, maintenance, and/or optimization.[4] (FF 11-16.) Therefore, Horman satisfies the accessing step of claim 1.

According to the discussion in the preceding paragraph, Horman also satisfies the "after generating" step of claim 1. The batch document (which includes several batch configuration files (note 4, *supra*)) remains accessible to synchronize the appended scripts, maintain, modify, or update the data and scripts, and optimize the connected administered servers, during and after testing. (FF 7-16.) Therefore, Horman also satisfies the last step of claim 1.

---

[4] The setup, update, and fix group batch files, and other batches, as test or production batches, are associated with each application (or application version), and remain accessible to the control server and any number of existing or added administered servers to either setup or update configurations and/or optimize data access. (FF 7-16.) According to Appellant's disclosure, the group batch document can be divided into sections. (FF 5.) Hence, Horman's setup, update, and fix files, as production or test batch files, each associated with a particular application, reasonably constitute different sections of one batch document, as set forth in claim 1.

Finally, the configuration files for each administered server have configuration information for, *inter alia*, the operating system, memory optimization, the data base instance, the data base manager, and the particular application. (FF 7, 9, 11.) For example, this information includes or pertains to "the database manager configuration parameter values, the data base design, and the database configuration parameter values." (FF 9.) Thus, the configuration files include "configuration information for one of a plurality of software components of the intelligent device," as set forth in claim 1.

Based on the foregoing discussion, Appellant has not demonstrated Examiner error in the rejection of claim 1, and claims 9, 10, 41, 48, 52, 66, and 68 which were not separately argued.

Claims 16, 23, and 24

Claim 16 recites accessing a batch configuration document comprising configuration information for the plurality of software components. As discussed above, Horman satisfies this accessing step. For example, Horman's PST template, having general configuration information for each software component, satisfies this step. Claim 16 also recites applying the configuration document to one or more configuration files. Applying the parameters of the template (batch configuration document) to respective sets of values in the relational tables for each administered server (configuration files) satisfies this step. (FF 9-15.)

Appellant argues (App. Br. 17) that because Horman teaches that a group batch is associated with a single end-user application, that Horman does not teach configuration information for a plurality of software components. The argument is unavailing. A group batch associated with

each administered server comprises configuration information for several software components, including the operating system, memory, and data base management and design, as described *supra*, with respect to claim 1. Each administered server has unique parameterized attributes and values associated with each different software component. That is, each application administers different - albeit similar - servers, each including a plurality of software components. (FF 7-15.)

Therefore, based on the discussion above, Appellant has not demonstrated Examiner error with respect to the rejection of claims 16, 23, and 24, the latter of which were not separately argued.

Claim 70

As Appellant notes (App. Br. 18), the Examiner rejected claim 70 based on rationale applied to claim 41. As noted *supra*, Appellant does not argue claim 41 separately from claim 1. Claim 70 includes elements similar to claim 41, and requires a carrier medium for implementing accessing a batch document and applying it to software components. Claim 70, also recites elements similar to method claim 16. (Appellant does not challenge the (known) carrier medium element of the claim.) Rather, Appellant's arguments (App. Br. 18-19) track those for claim 16. Therefore, for reasons similar to those outlined *supra*, Appellant fails to demonstrate error.

Claims 4, 33, and 37

Claims 4 and 33 include the limitation of configuring a second intelligent device by transferring and/or using the batch configuration document generated on the first intelligent device. Appellant argues (App. Br. 19, 23-24) that Horman does not teach using, or transferring, the batch configuration document, on, or to, a second intelligent device. Horman

teaches that the batch document is scalable to configure hundreds to thousands of servers (FF 8) and is used typically to install (setup) administered servers (FF 15, 16). Another such administered server constitutes a second intelligent device. Horman's additional intelligent servers locally execute (FF 13, 14) the batch document generated on the first intelligent device (e.g., as described with respect to claim 1), thereby satisfying claims 4 and 33. That is, this local execution on the second intelligent administered server satisfies the configuring step of claim 33 and the transferring step of claim 4.

Under an alternative interpretation, copying the batch document generated on an administrative control server and transferring it to another intelligent control server meets claims 4 and 33. Appellant admits that transferring configuration files was known. (FF 4.) Horman also teaches copying prior batches to incorporate modifications on test servers, and then applying those batches to production servers. (FF 16.) In light of these teachings, copying programs, including a batch document, from one intelligent device to another, would have been a routine matter of common knowledge and common sense, for example, as a back-up to mitigate damage caused by Horman's original control server crashing or becoming obsolete. "'Our suggestion test is in actuality quite flexible and not only permits, but *requires*, consideration of common knowledge and common sense.'" *KSR,* 550 U.S at 421 (citation omitted). Such a step of transferring or using amounts to employing a known step to perform its predictable function.

Appellant also presents (App. Br. 19) arguments similar to those for claim 1, arguing that Horman's scripts do not satisfy the batch document

generation as required by claim 33. As discussed above, Horman's batches comprise scripts (*see e.g.* FF 14), and Horman generates the batch document by accessing configuration files. Accordingly, for the reasons discussed above, Appellant fails to demonstrate error with respect to the rejection of claims 4 and 33, and also, claim 37 which was not separately argued.

Claims 56 and 60

Appellant states (App. Br. 20) that replacing parameters of sync scripts with particular values does not constitute applying configuration information from a batch document to configuration files located on the intelligent device. However, Appellant fails to explain why this sync script replacement does not meet the argued limitation. For reasons similar to those explained above with respect to claim 1, these particular values constitute information from the segmented batch file document and its template which are applied to particular configuration files (in order to optimize the various software components). Appellant's remaining arguments (App. Br. 20-21) track those for the claims above, and fail to demonstrate error in the rejection of claim 56, and claim 60 which was not separately argued.

Claims 2 and 18

Claims 2 and 18 require using a set of executable instructions, termed a script, to execute, respectively, various steps of claims 1 and 16. Horman's computer system uses a set of instructions to carry out the steps outlined in claims 1 and 16 as implied in the discussions above. For example, based on an authentication "script," the control server determines which batch steps (i.e., which configuration file(s)) to select and execute on each administered server. (*See* FF 14.) Appellant does not define a script

nor argue why a certain set of implied instructions does not constitute a script.

Rather, Appellant's arguments (App. Br. 22, 29-30) focus on Horman's synchronization scripts. Those scripts, employed in the batch file document and configuration files, are a mere subset of the various sets of instructions employed by Horman's system to carry out the batch process. In other words, setting aside any confusion involving Horman's various scripts discussed *supra*, Horman reasonably teaches another set of instructions, reasonably constituting the claimed "script," to perform the claimed steps. Appellant's remaining arguments (*id.*) track those for claim 1. As such, Appellant fails to demonstrate error.

Claims 3 and 8

Claim 3 calls for configuring software components by setting configuration information in the files before accessing them. Assuming arguendo that the claimed files can have information placed in them before accessing them, claim 3 is broad enough to encompass accessing the files a first time at setup to put information in them before accessing the files a second time, either at the appended setup, or at maintenance, or testing, as Horman's system does. (*See* the claim 1 discussion *supra*; note 4; and FF 15-16.) This claim interpretation is reasonable, given that claim 1, from which claim 3 depends, implies two accessing steps: first, "accessing" and second, "after said generating, the batch configuration document is accessible."

Claim 8 recites an initializing step. Appellant does not challenge the Examiner's finding that "in order to execute a software, it must be initialized." (App. Br. 25 (quoting the Examiner) (citation omitted).)

Appellant generally asserts more than initializing is required and recites the claim language. Claim 8 does require that the configuration files use the initializing information. Similar to the analysis above for claim 3, Horman's system employs specific configuration files having specific initial populated values for each server at setup, thereby meeting the initializing step. (FF 11, 12, 15.)

Claim 5

Appellant relies (Reply Br. 13) on arguments similar to those presented for claim 1, and asserts that the configuration information does not apply to another device. Based on the discussion above with respect to claims 1, 4 and 33, any of the added servers constitute another intelligent device with software thereon employing the batch documents of the intelligent device of claim 1. (FF 8, 15.)

Claim 7

Claim 7 requires applying the batch document to one or more configuration files associated with one or more software devices. Horman's system performs this, as explained above with respect to claims 1 and 16. (FF 11-16.) Appellant's arguments (App. Br. 24-25) do not demonstrate error.

Claim 11

Claim 11 calls for the plurality of software components to include software drivers for hardware components. Appellant does not dispute (Reply Br. 16) the Examiner's finding that software drivers, to enable hardware to perform, were well known. Appellant asserts (Reply Br. 16) that the record does not suggest that the known software drivers are "configurable according to the limitations recited in claim 1, from which

claim 11 depends." However, Appellant admits that such configurable drivers fall under the category of related art, which is assumed here to be admitted prior art, given the related discussion of admitted prior art Figure 1 under the same section heading. (*See* FF 3, 4; Spec. 1-4 ("Description of the Related Art").)

Under these circumstances, substituting a particular configurable driver software component for one of the myriad of configurable software components of hardware such as the administrative server or implied memory hardware, as taught by Horman (FF 11, 12), constitutes the predicable use of a prior art element to perform its known function. Hence, Appellant fails to demonstrate error in the rejection of claim 11.

Claims 12, 38, 45, 53, and 61

Appellant's argument (Reply Br. 16-17) focuses on claim 12, hereby selected to represent the group. The Examiner relies on Horman at column 8, lines 14-16, to teach configuration files which include operating system configuration information. Appellant does not respond to this, but instead, focuses on a previous citation (to Horman at column 6) by the Examiner. (*See* Reply Br. 16-17.) Horman discusses scripts containing software operating system configuration commands at the passage the Examiner relies upon. (*See* FF 11.) As such, Appellant fails to demonstrate error in the rejection of these claims.

Claims 13, 14, 27, 28, 35, 46, 54, 62, 69, and 72

Appellant asserts (Reply Br. 17-18) that the Examiner provides no reason for substituting the Markup Language (extensible Markup Language (XML)) format, as set forth in claim 13, for the language format of Horman. Based on Appellant's arguments directed to claim 13, claim 13 is selected as

representative of this group. As Appellant's state, the Examiner reasoned (Ans. 25) that the XML format was known for configuration files and that batch configuration files were known in the art; and therefore, it would have been obvious to make the substitution as a matter of design choice.[5] Contrary to Appellant's argument, this "mere substitution," *KSR*, 550 U.S. at 416, rationale suffices to establish a prima facie case of obviousness. Appellant merely substitutes one programming language for another. Appellant's denials fail to rebut the Examiner's prima facie case.

Claims 15, 29, 36, 47, 55, 63, 69, and 72[6]

Appellant similarly asserts (Reply Br. 18-19) that the cited art does not teach an XML Document Type Definition (DTD) format for batch documents as set forth in claim 15, hereby selected as representative based on Appellant's arguments. The Examiner found (Ans. 9) that Mossman teaches the particular format for configuration files at paragraph 0091. Mossman refers there to an XML format, but does not recite the particular XML DTD format. However, Appellant does not explain how the two formats differ. Appellant asserts a lack of a reason for making the substitution.

Contrary to Appellant's argument, this "mere substitution," *KSR*, 550 U.S. at 416, rationale suffices to establish a prima facie case of obviousness. Appellant merely substitutes one programming language for another. Appellant does not argue that this mere substitution would have been "uniquely challenging or difficult for one of ordinary skill in the art" or

[5] Evidence of record supports the Examiner's finding that using different programming languages involves routine matters of design choice. (FF 17.)
[6] Appellant repeats (Reply Br. 17-18) claims 69 and 72 and indicates the claims are grouped with claims 13 and 15.

"represented an unobvious step over the prior art." *Leapfrog*, 485 F.3d at 1162. Instead, Appellant relies on arguments similar to those presented for claim 13. Under these circumstances, Appellant has not met the burden of demonstrating error in this "mere substitution," *KSR*, 550 U.S. at 416. (*See also* note 5 *supra*.)

Claims 6, 19-22, 50, 57-59, 67, and 71

Similar to the analysis for claims 13 and 15 *supra*, this analysis requires determining whether paragraphs in another teaching, i.e., Shafron, describing scripts that may be used to manipulate a Document Object Model (DOM) (according to Appellant's characterization (Reply Br. 28) of the teaching), satisfy the limitation of "generating a Document Object Model (DOM) tree from each of a plurality of accessed configuration files," as set forth in claim 6. Appellant raises (*id.*) similar issues with respect to the claims listed under this heading. Claim 6 is selected as representative based on the similar scopes and arguments.

The Examiner does point to particular Shafron paragraphs (Ans. 15) to teach the DOM format, but no mention of DOM trees appear there. On the other hand, the Examiner does articulate (Ans. 15) a reason to create trees: "one would have been motivated to generate a DOM tree because DOM allows programs and scripts to access and update the content, structure and style of documents dynamically."

Appellant does not challenge this statement with sufficient persuasiveness or particular emphasis on the trees as being significantly different than a DOM format or language. Shaffron teaches that different language and language manipulations are routine matters of design choice. (FF 17.) Contrary to Appellant's argument, this at least constitutes a "mere

substitution," *KSR*, 550 U.S. at 416, rationale which suffices to establish a prima facie case of obviousness. Appellant merely substitutes one programming language for another or generates a particular data format (DOM tree).

A DOM tree appears to be a format or structure for representing data. Appellant does not argue that this mere substitution of using DOM trees to access data for Horman's program language employing "schema, tables, indexes, and any other data based objects" (FF 15) would have been "uniquely challenging or difficult for one of ordinary skill in the art" or "represented an unobvious step over the prior art." *Leapfrog,* 485 F.3d at 1162. Thus, Appellant fails to establish error.

Claim 17

Appellant argues (App. Br. 17) that Horman's system of replacing parameters in a parameterized script template does not meet the step of replacing one or more current parameter values in the particular configuration file with new parameter values from the batch configuration document. Claim 17 depends from claim 16. Under the analysis of claim 16 above, Horman's specific parameter values within the configuration files replace parameters in the batch document template at setup. (*See* FF 12, 15.) The Examiner relies (Ans. 9) on column 2, lines 50-65 of Horman to teach the added limitations of claim 17.

That Horman column generally teaches, *inter alia*, that "the parameters allow for the existence of a general description of administrative action of the group (the PST) by abstracting away the differences among the members of the group." (FF 12.) These differences exist among thousands of machines (FF 7, 8), and Horman's system employs (FF 15) "[b]atches . . .

to ensure that the administered servers in a group remain as similar as possible." Hence, Horman suggests replacing specific (and generic) parameterized values within each configuration file with generic values so that members within a group generally use the same, or similar, parameter values. Horman accomplishes this goal by using appended setup batch steps, or modifying successive test batches and thereafter providing data updates in an update batch. (FF 15-16.) These appended setup batch steps and data updates, or modified test batches "constitute configuration information from the batch configuration document" which serve to "replace one or more current parameter values in the particular configuration file" associated with a particular administered server prior to the appended batch step and/or data update. Therefore, Appellant fails to demonstrate error.

Claims 30, 39, and 64

Claim 30 recites a rebooting limitation wherein after applying the configuration information to the files, rebooting applies the configuration information to one or more software devices. Appellant focuses (App. Br. 30-31) on claim 30, hereby selected as representative. Appellant does not dispute the Examiner's finding that "'for any change or configuration to take place, the computer must be rebooted.'" (App. Br. 31 (quoting the Examiner) (citation omitted).) Appellant argues (*id.*) that the Examiner has not cited anything in the combined references that teaches all the specific limitations in claim 30. However, Appellant does not point out what limitations are missing from the combination. Horman meets all the elements of claim 16, from which claim 30 depends, and claim 30 reasonably only adds applying a rebooting step to instigate the configuration as recited in claim 16. No dispute exists about whether the step is known.

Appellant's additional argument (App. Br. 30) that Mossman's teachings directed to virtual re-booting teach away from the claimed invention fail to demonstrate error, because, given the known step of re-booting to configure or re-configure software, skilled artisans would not have been "discouraged from following the path set out," *In re Gurley*, 27 F.3d at 553. That Mossman arguably teaches a better or different way does not constitute a teaching away. Therefore, Appellant has not demonstrated error in the rejection of claims 30, 39, and 64.

Claims 31 and 65

Representative claim 31 requires initializing the software components. Horman's setup batch system initializes (at setup) the software components in the manner required by these claims, according to the discussion of claims 3 and 8 above. (*See also* FF 15.) Therefore, Appellant has not demonstrated error in the rejection of these claims.

Claim 32

Claim 32 requires "generating the batch configuration document on a different intelligent device prior to said accessing" the batch configuration document of claim 16. Assuming for the sake of argument that a system can generate a document prior to accessing it, claim 32 apparently requires generating the batch document on one intelligent device, and then accessing it on another device. Connecting an additional (accessing) administered intelligent server to the different (generating) intelligent device of claim 16, as Horman teaches (FF 8, 16), satisfies claim 32. Also, downloading a new tested batch version from a test server, containing remnants of the old version, to an existing server, as Horman teaches (FF 16), also meets the claim. (*See also* the related discussion *supra* with respect to claims 4 and

33, incorporated here.) Therefore, Appellant has not demonstrated error in the rejection of claim 32.

Claims 34 and 49

Appellant's arguments (App. Br. 32-33, 35) do not demonstrate error. Claim 34 reasonably requires either transferring Horman's group batch file from the original control server to a second intelligent new control server, adding another intelligent administered server, or downloading a tested version containing remnants of an older version. Claim 49 recites a similar limitation. Therefore, according the discussion *supra* involving claims 4, 32, and 33, Horman satisfies claims 34 and 49.

Claim 40

Appellant relies (App. Br. 33) on arguments for claim 33 which are unavailing for the reasons discussed *supra*. Appellant also argues (*id.*) that Horman does not teach storing the generated batch configuration document on a server, wherein the server is coupled to the second intelligent device via a network. Horman's control server, the first intelligent device, stores the batch file (FF 12), and is coupled to any one of many second intelligent devices, as discussed above with respect to claim 33, as required by claim 40. Therefore, Horman satisfies claim 40.

Claim 42

Notwithstanding Appellant's arguments (App. Br. 33-34), as discussed *supra*, Horman teaches adding additional administered servers (FF 8, 16) and rendering the batch document accessible to these additional intelligent devices (containing operating systems), thereby satisfying the disputed elements of claim 42. Under another alternative discussed *supra*, transferring a batch file document to other administrative servers (and group

of administered servers) after testing the document would have been an obvious modification.

Claims 43, 44, 51, and 52

Appellant's arguments (App. Br. 34-35) do not demonstrate any particular error. Reasonably similar corresponding limitations, at least with respect to representative claim 43, appear in claims 1, 7, and 16, which have been addressed *supra*. While Appellant notes (*id.*) that claim 3 recites different limitations, claim 3 depends from claim 1. Appellant's arguments do not point to any missing limitations with the particularity required to establish error.

Claims 25 and 26

Appellant does not present any patentability arguments for these claims. Therefore, Appellant has waived any potential arguments for claims 25 and 26 and has failed to demonstrate error.

CONCLUSION

Appellant did not show that the Examiner erred in finding that Horman and Mossman collectively teach accessing a plurality of configuration files on an intelligent device generating a batch configuration document from the plurality of configuration files as set forth in claim 1. Appellant did not show that the Examiner erred in finding that Horman, Mossman, and Shafron collectively teach the elements of the remaining claims on appeal.

DECISION

We affirm the Examiner's decision rejecting claims 1-72.

No time period for taking any subsequent action in connection with this appeal may be extended under 37 C.F.R. § 1.136. *See* 37 C.F.R. § 1.136(a)(1)(iv).

AFFIRMED

KMF

MHKKG/SUN
P.O. Box 398
Austin, TX 78767